# On the Power of Las Vegas for One-Way Communication Complexity, OBDDs, and Finite Automata

Juraj Hromkovič[1]

*Lehrstuhl für Informatik I, RWTH Aachen, Ahornstraße 55, 52 074 Aachen, Germany*

and

Georg Schnitger

*Fachbereich Informatik, Johann Wolfgang Goethe-Universität Frankfurt, Robert Mayer Straße 11-15, 60054 Frankfurt am Main, Germany*

Received January 31, 2001

The study of the computational power of randomized computations is one of the central tasks of complexity theory. The main goal of this paper is the comparison of the power of Las Vegas computation and deterministic respectively nondeterministic computation. We investigate the power of Las Vegas computation for the complexity measures of one-way communication, ordered binary decision diagrams, and finite automata.

(i) For the one-way communication complexity of two-party protocols we show that Las Vegas communication can save at most one half of the deterministic one-way communication complexity. We also present a language for which this gap is tight.

(ii) The result (i) is applied to show an at most polynomial gap between determinism and Las Vegas for ordered binary decision diagrams.

(iii) For the size (i.e., the number of states) of finite automata we show that the size of Las Vegas finite automata recognizing a language $L$ is at least the square root of the size of the minimal deterministic finite automaton recognizing $L$. Using a specific language we verify the optimality of this lower bound. 

*Key Words*: computational complexity; Las Vegas; determinism; nondeterminism; communication complexity; finite automata; ordered binary decision diagrams.

## 1. INTRODUCTION AND DEFINITIONS

The comparative study of the computational power of nondeterministic, deterministic, and randomized computations is one of the central tasks of complexity theory. In this paper we focus on the relationship between Las Vegas and determinism and between Las Vegas and nondeterminism.

The relationship between the complexity classes P and ZPP, the class of languages accepted by polynomial-time Las Vegas Turing machines, is unresolved. One can consider counterparts of these classes for other computing models. In [1] such counterparts for communication complexity have been introduced. Already in 1982, Mehlhorn and Schmidt [2] proved an at most quadratic gap between determinism and Las Vegas for communication complexity. Similar polynomial gaps between Las Vegas and determinism are known for the combinational complexity of non-uniform circuits, the space complexity of Turing machines [3], and the time complexity of CREW PRAMs [4].

Generally, for fundamental computing models, one conjectures that the costs of Las Vegas computations are closer to the costs of deterministic computations than to the costs of nondeterministic computations.

We investigate the relationships between determinism, Las Vegas, and nondeterminism for the following three complexity measures:

[1] The work of this author has been supported by the DFG Project HR 14/3-2.



0890-5401/01 $35.00

(i) message length in one-way communication complexity,

(ii) the size of ordered binary decision diagrams (OBDDs),

(iii) the size (i.e., the number of states) of finite automata.[2]

To define Las Vegas computations we follow [5] and consider *self-verifying nondeterminism*.[3] A self-verifying nondeterministic machine $M$ is allowed to give three possible answers "yes," "no," "don't know." $M$ is not allowed to make mistakes: If the answer is "yes," then the input must be in $L(M)$. If the answer is "no," then the input cannot be in $L(M)$. For every input there is at least one computation that does not finish with the answer " don't know."

We say that $M$ is a Las Vegas machine recognizing a language $L$ if and only if $M$ is a self-verifying nondeterministic machine recognizing $L$ and for each input the answer "don't know" is given with probability at most $\frac{1}{2}$. Note that this upper bound $\frac{1}{2}$ on the failure probability (unsuccess) is not essential. Any $\varepsilon < 1$ can be considered.

Obviously, the difference between self-verifying nondeterminism and nondeterminism is that the negative answer of a nondeterministic machine gives no information about the relationship of the input to $L(M)$. The answer "no" only means that the machine has not succeeded in proving that the input is in $L(M)$. On the other hand, for self-verifying nondeterminism the answer "no" implies that $M$ has proved in this computation that $x \notin L(M)$.

We observe that the concept of self-verification is general and can be applied to almost all computing models. The consideration of the intersections of some complexity classes with their corresponding complement classes (like NP $\cap$ co-NP) is only a special case of this concept. Self-verification allows a natural definition of Las Vegas computation as a restricted self-verifying nondeterministic computation. And this restriction is the same as the restriction one uses to define Monte Carlo computation as a restricted nondeterministic computation. So, the difference between Las Vegas and Monte Carlo may be considered to be of the same nature as the difference between self-verifying nondeterminism and nondeterminism.

Self-verifying nondeterminism is of independent interest, since it provides a natural concept for a comparative study of the complexities of solution verification, search for a solution, and proving the nonexistence of solutions:

1. The complexity of deterministic computations is the maximum of {complexity of the search for a solution, complexity of finding a proof of the nonexistence of any solution}.

2. The complexity of nondeterministic computations is the complexity of verifying that a guessed candidate for a solution is a correct solution.

3. The complexity of self-verifying nondeterminism is the maximum of {complexity of verification of a guessed candidate, complexity of verifying a guessed proof of the nonexistence of any solution}.

We compare deterministic and nondeterministic computations not only with Las Vegas computations, but also with general self-verifying nondeterministic computations. The main results of this paper are as follows:

(i) *One-way communication complexity*: We consider the one-way version of two-party protocols as introduced by Yao [6] for a fixed partition of the input. Computer $C_I$ receives the first half of the input and computer $C_{II}$ receives the rest. Informally, a deterministic one-way protocol $P$ first determines the message sent from computer $C_I$ to computer $C_{II}$ on the basis of the input of $C_I$ and then decides whether $C_{II}$ accepts or rejects. The decision of $C_{II}$ is made solely on the basis of the received message and the input of $C_{II}$.

The *one-way communication complexity of* $P$, $cc_1(P)$, is the length of the longest message sent by $C_I$. Finally for a Boolean function $f$, $\mathrm{cc}_1(f)$ denotes the one-way communication complexity of the best protocol computing $f$. Let $\mathrm{ncc}_1(f)$ (resp. $\mathrm{svncc}_1(f)$, $\mathrm{lvcc}_1(f)$) denote the one-way nondeterministic (resp. self-verifying, Las Vegas) communication complexity of $f$.[4]

[2] Note that space is the logarithm of the size of finite automata.

[3] Note that elsewhere the notion strong nondeterminism instead of self-verifying nondeterminism has been used.

[4] For formal definitions and details see the monographs on communication complexity [7, 8].

In our main result of this part we show that

$$\mathrm{lvcc}_1(f) \geq \mathrm{cc}_1(f)/2 \qquad (\star)$$

for every Boolean function $f$. This result is quite surprising, because there is a quadratic gap between Las Vegas and determinism for the general (two-way) model of communication complexity [2]. Moreover, for a specific language $L \subseteq \{0,1\}^*$, we show that $\mathrm{cc}_1(h_n(L)) = 2 \cdot \mathrm{lvcc}_1(h_n(L))$, where $h_n(L)$ is the Boolean function of $n$ variables defined by $h_n(L)(\alpha) = 1$ iff $\alpha \in L \cap \{0,1\}^n$. Hence the relationship in $(\star)$ is best possible.

Note that the constant 2 in the relationship $(\star)$ is the consequence of choosing $\frac{1}{2}$ as the bound on the probability to finish the communication with the answer "don't know." If one considers an arbitrary $\varepsilon < 1$ instead of $\frac{1}{2}$, then $(\star)$ has to be replaced by $\mathrm{lvcc}_1(f) \geq (1-\varepsilon) \cdot \mathrm{cc}_1(f)$.

It is well known that there exist languages $A$ with an exponential gap between $\mathrm{cc}_1(h_n(A))$ and $\mathrm{ncc}_1(h_n(A))$. Using a simple, standard argument we show that there is a language $L$ with an exponential gap between $\mathrm{cc}_1(h_n(L))$ and $\mathrm{svncc}_1(h_n(L))$. Thus, self-verifying nondeterminism may be much more powerful than determinism. This establishes another substantial difference between one-way and two-way communication complexity, where self-verifying nondeterminism is polynomially related to determinism [9, 10].

(ii) *Ordered binary decision diagrams*: OBDDs [11, 12] are a restricted version of branching programs [13, 14] and are successfully used as data structures for Boolean functions. They allow succinct representation of many Boolean functions and possess efficient algorithms for the most important operations on this data structure. The complexity of an OBDD $A$, called the size of $A$, is the number of nodes of $A$. Previously only Monte Carlo randomized OBDDs and branching programs have been investigated (see, e.g., [15]). Las Vegas randomized OBDDs are considered here for the first time. Applying the results of (i) for one-way communication complexity we prove a polynomial relation between determinism and Las Vegas for the size of OBDDs.

(iii) *Finite automata*: We consider the model of one-way finite automata. In the following $L(A)$ denotes the language accepted by the computing device $A$. We also consider self-verifying nondeterministic finite automata (SNFA) as nondeterministic automata whose states are partitioned into three disjoint groups: accepting states, rejecting states, and neutral states. An input word is accepted (rejected) by a SNFA if there exists a computation finishing in an accepting (rejecting) state. Moreover, for no input there exist two computations finishing in the accepting and rejecting state, respectively, and for each input at least one computation is accepting or rejecting.

We introduce a Las Vegas finite automaton (LVFA) as a SNFA $A$ which for any $x \in L(A)$ reaches an accepting state with probability at least $\frac{1}{2}$ and which for any $x \notin L(A)$ reaches a rejecting state with probability at least $\frac{1}{2}$. For every state $q$ of $A$ and every symbol $a$ of the input alphabet, we allow an arbitrary probability distribution over the set of edges leaving $q$ and labelled by $a$. The probability of a computation of $A$ is the product of the transition probabilities along the path of the computation.

For any regular language $L$ we define $\mathrm{s}(L)$, $\mathrm{ns}(L)$, $\mathrm{svns}(L)$, and $\mathrm{lvs}(L)$ respectively as the size of a minimal deterministic, nondeterministic, self-verifying nondeterministic, and Las Vegas finite automaton for $L$.

The main result of this part shows that

$$\mathrm{lvs}(L) \geq \sqrt{\mathrm{s}(L)}$$

for every regular language $L$. The optimality of this lower bound on $\mathrm{lvs}(L)$ is verified by constructing a regular language $L'$ with $\mathrm{s}(L') = \Omega((\mathrm{lvs}(L'))^2)$. Again, the result $\mathrm{lvs}(L) \geq \sqrt{\mathrm{s}(L)}$ is connected with bounding the probability of reaching a neutral state by $\frac{1}{2}$. If one takes an arbitrary $\varepsilon < 1$ instead of $\frac{1}{2}$, then one obtains $\mathrm{lvs}(L) \geq (\mathrm{s}(L))^{1-\varepsilon}$.

It is well known that there are regular languages with $\mathrm{s}(L) \sim 2^{\mathrm{ns}(L)}$. Here, we show that for some regular languages $A$, $B$ there are exponential gaps between $\mathrm{s}(A)$ and $\mathrm{svns}(A)$ and between $\mathrm{svns}(B)$ and $\mathrm{ns}(B)$.

Recently some of our results have been generalized to quantum communication in [16]. It is shown there that even quantum Las Vegas one-way protocols cannot be better than deterministic one-way

protocols for total functions by more than a factor of 2. Our result about Las Vegas automata also generalizes to the quantum case.

This paper is organized as follows. In Section 2 we analyze the Las Vegas communication complexity and apply the obtained results in Sections 3 and 4 to OBDDs and automata respectively.

## 2. ONE-WAY COMMUNICATION COMPLEXITY

We consider one-way Las Vegas protocols with a public[5] random source [17]. Note that this strengthens the lower bound results because the lower bounds on randomized protocols with public random sources are also lower bounds on randomized protocols with private random sources. Let, for any function $f : X \times Y \to \{0, 1\}$,

$$M(f) = [a_{u,v}]_{u \in X, v \in Y} \quad \text{with } a_{u,v} = f(u, v)$$

be the *communication matrix of f*.

First, we present our main result.

THEOREM 2.1. *For every function* $f : X \times Y \to \{0, 1\}$ *with finite sets* $X$ *and* $Y$,

$$\text{lvcc}_1(f) \geq \text{cc}_1(f)/2.$$

*Proof.* First, we give an informal idea of the proof. Let $f : X \times Y \to \{0, 1\}$ be a finite function. We represent $f$ by its communication matrix $M(f) = [a_{u,v}]_{u \in X, v \in Y}$ with $a_{u,v} = f(u, v)$. The number of different messages of an optimal one-way protocol $P$ computing $f$ is exactly the same as the number *row*$(M(f))$ of different rows of $M(f)$; i.e., $\text{cc}_1(f) = \lceil \log_2 \text{row}(M(f)) \rceil$ [9].

Any one-way Las Vegas protocol $P'$ may be considered as a collection of deterministic one-way protocols $P_1, P_2, \ldots$ with probabilities $p_1, p_2, \ldots$.[6] For any input $\alpha$, $P_i$ may produce the results 0,1 or $*$ (i.e., "don't know"). Since $P'$ is a Las Vegas protocol, no protocol $P_i$ ever errs and for every $(u, v) \in X \times Y$, the protocols $P_1, P_2, \ldots$ produce the output $*$ with probability at most $\frac{1}{2}$. To any protocol $P_i$ $(i = 1, 2, \ldots)$, one can assign its $0/1/*$ communication matrix $M(P_i) = [b^i_{uv}]_{u \in X, v \in Y}$, where $b^i_{uv} = a_{uv}$ if $P_i$ does not give output $*$ and $b^i_{uv} = *$ otherwise.

Our main goal is to find one protocol $P_i$ such that $M(P_i)$ has at least $\sqrt{\text{row}(M(f))}$ different rows. In order to reduce the number of different rows of these deterministic protocols we will have to replace certain entries of $M(f)$ by a $*$ in a clever way. Obviously, replacing certain entries of $M(f)$ by $*$ will help reduce the number of different rows far more than the replacement of other entries by $*$. For the identity matrix the diagonal entries play this helper role. For instance we can reduce the number of different rows to two by setting the upper left and the lower right quarter to $*$. Observe that this radical reduction in the number of different rows is obtained after replacing only one half of the entries by $*$. On the other hand, any significant reduction in the number of different rows has to involve the diagonal entries and any such entry has to stay untouched with probability at least one half. An obvious averaging argument shows that one deterministic protocol exists with at least $N/2$ different rows (if we consider the $N \times N$ identity matrix).

In the above example the diagonal entries form a fooling set and any Las Vegas communication protocol has to have at least $\frac{|F|}{2}$ messages for a fooling set $F$. However, we cannot expect to find large fooling sets in general. In particular, the $n \times \log_2 n$ communication matrix $M^*$ whose $i$th row contains the binary representation of $i$ possesses only fooling sets of logarithmic size. But it can be shown in this case that any Las Vegas one-way protocol has to have $\sqrt{n}$ messages.

Our proof will introduce a new notion of fooling sets. Set $M(f) = M$ and assume that $M$ has $r$ pairwise different rows and $c$ pairwise different columns. Our new notion of fooling sets is based on a real-valued weight assignment

$$\text{weight} : \{1, \ldots, r\} \times \{1, \ldots, c\} \to \mathbb{R}$$

[5] Sometimes also called common random source.

[6] This follows, since we consider Las Vegas protocols with a public random source.

for $M$. Let $I = \{1, \ldots, r\}$. We define the function weight recursively, processing $M$ column after column in a clever way.

*Case 1.* If column 1 is monochromatic for all rows in $I$, then set, for $i \in I$,

$$\text{weight}(i, 1) = 0$$

and $I = I - \min\{i \in I\}$.

*Case 2.* If column 1 of $M$ is not monochromatic for the rows in $I$, then there is a $d$, with $0 < d < 1$, such that $d \cdot |I|$ rows have a 0 in column 1 (and $(1-d) \cdot |I|$ rows have a 1 in column 1). We set

$$\text{weight}(i, 1) = \begin{cases} \log_2\left(\frac{1}{d}\right) & \text{if } M[i, 1] = 0 \\ \log_2\left(\frac{1}{1-d}\right) & \text{otherwise,} \end{cases}$$

and define

$$I_0 = \{i \in I \mid M[i, 1] = 0\} \qquad \text{and} \qquad I_1 = \{i \in I \mid M[i, 1] = 1\}.$$

The procedure recursively continues with $I_0$ and $I_1$. Observe that the procedure stops if the row sets are singletons, since then all columns will be monochromatic.

We begin our analysis with the following technical fact. In what follows $0 \cdot \log 0$ is defined to be 0.

FACT 2.1. *For any $x, y \geq 0$ and $d \in (0, 1)$,*

$$x \cdot \log_2 \frac{x}{d} + y \cdot \log_2 \frac{y}{1-d} \geq (x + y) \cdot \log_2(x + y).$$

*Proof.* The cases where $x$ or $y$ are 0 are trivial. Write $p = x/(x + y)$ and $q = y(x + y)$. Then $p + q = 1$. The fundamental fact that the informational divergence

$$\sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

for any two probability distributions $(p_1, \ldots, p_n)$ and $(q_1, \ldots, q_n)$ is always nonnegative [18] tells us that

$$p \cdot \log(p/d) + q \cdot \log(q/(1-d)) \geq 0$$

for every $d \in (0, 1)$. Adding the obvious equality (remember that $p + q = 1$)

$$p \cdot \log(x + y) + q \cdot \log(x + y) = \log(x + y)$$

and cancelling yields

$$p \cdot \log(x/d) + q \cdot \log(y/(1-d)) \geq \log(x + y).$$

Multiplying by $x + y$ yields the claimed inequality. ■

*Proof of Theorem 2.1 continued.* For a subset $R \subseteq \{1, \ldots, r\}$ set

$$\text{differ}(R) = \{j \mid \exists i_1, i_2 \in R : M[i_1, j] \neq M[i_2, j]\}.$$

Now, we are ready to analyze the properties of our weight assignment.

LEMMA 2.2. (a) *For each* $(i, j) \in \{1, \ldots, r\} \times \{1, \ldots, c\}$,

$$\text{weight}(i, j) \geq 0.$$

(b) *For each* $i \in \{1, \ldots, r\}$,

$$\sum_{j=1}^{c} \text{weight}(i, j) = \log_2 r.$$

(c) *For any* $R \subseteq \{1, \ldots, r\}$,

$$\sum_{j \in \text{differ}(R)} \sum_{i \in R} \text{weight}(i, j) \geq |R| \cdot \log_2 |R|.$$

*Proof.* Part (a) is immediate by construction. We verify part (b) by induction on $r$. The basis for $r = 1$ is trivial. For the inductive step we can assume without loss of generality that column 1 is not monochromatic. Let $I_0$ (resp. $I_1$) be the set of those rows with a zero (resp. one) in column 1 and assume that $|I_0| = c \cdot r$.

We apply the induction hypothesis to the rows in $I_0$ and $I_1$. For a row $i \in I_0$ we obtain

$$\sum_{j=2}^{c} \text{weight}(i, j) = \log_2(c \cdot r).$$

But weight $(i, 1) = \log_2(\frac{1}{c})$ and

$$\sum_{j=1}^{c} \text{weight}(i, j) = \log_2 \left(\frac{1}{c}\right) + \log_2(c \cdot r) = \log_2 r.$$

Part (b) follows with a symmetric argument for the rows in $I_1$.

We apply induction on the size of $R$ to verify part (c). The basis for $|R| = 1$ is again trivial. We assume for the inductive step that column 1 is not monochromatic for the rows in $R$. Hence $R$ splits into the subsets $R_0$ resp. $R_1$ of those rows in $R$ with value zero (resp. one) in column 1. Since we can apply the induction hypothesis to $R_0$ and $R_1$, we obtain

$$\begin{aligned}\sum_{j \in \text{differ}(R)} \sum_{i \in R} \text{weight}(i, j) &= \sum_{i \in R} \text{weight}(i, 1) + \sum_{j \in \text{differ}(R),\, j \neq 1} \sum_{i \in R} \text{weight}(i, j) \\ &\geq |R_0| \cdot \log_2 \left(\frac{1}{c}\right) + |R_1| \cdot \log_2 \left(\frac{1}{1-c}\right) \\ &\quad + |R_0| \cdot \log_2(|R_0|) + |R_1| \cdot \log_2(|R_1|).\end{aligned}$$

Thus part (c) follows from Fact 2.1. ■

*Proof of Theorem 2.1 continued.* Assume we have a one-way Las Vegas protocol $P'$ for a Boolean function $f$ represented by the matrix $M(f)$ with $r$ pairwise different rows. Let the function weight be defined for $M(f)$ with the above three properties. Then there is a deterministic one-way protocol $P \in \{P_1, P_2, \ldots\}$ such that

$(*)$ the sum of all weights of entries of $M(P)$ with value $*$ is at most one half of the sum of all weights (i.e., at most $\frac{1}{2} \cdot \sum_{i=1}^{r} \sum_{j=1}^{c} \text{weight}(i, j) = \frac{r}{2} \cdot \log_2 r$ according to property (b) of the weight assignment).

This follows, since for every input the output of $P'$ is equal to $*$ with probability at most one half. The deterministic protocol $P$ partitions the set of all rows of $M(f)$ into classes $R_1, \ldots, R_k$ of identical rows

(after replacing certain entries by $*$). By Lemma 2.2 (c) of the function weight we obtain for any class $R_s$

$$\sum_{j \in \text{differ}(R_s)} \sum_{i \in R_s} \text{weight}(i, j) \geq |R_s| \cdot \log_2 |R_s|.$$

Let $M(R_s)$ be the restriction of $M(f)$ to the rows in $R_s$. The quantity on the left-hand side is a lower bound for the weight of all entries of $M(R_s)$ with value $*$. Since the function $\sum_{s=1}^{k} x_s \log_2 x_s$ with $x = \sum_{s=1}^{k} x_s$ is minimized for $x_1 = \cdots = x_k = \frac{x}{k}$,

$$\sum_{s=1}^{k} x_s \log_2 x_s \geq k \cdot \left( \frac{x}{k} \cdot \log \frac{x}{k} \right) = x \log_2 x - x \log_2 k.$$

Hence the sum of weights of entries of $M(P)$ with value $*$ is at least

$$\sum_{s=1}^{k} |R_s| \log_2 |R_s| \geq r \log_2 r - r \cdot \log_2 k.$$

From the above inequality and from $(*)$, it follows that

$$\frac{r \log_2 r}{2} \geq \sum_{s=1}^{k} |R_s| \log_2 |R_s| \geq r \log_2 r - r \cdot \log_2 k$$

and hence that $k \geq \sqrt{r}$. In other words, $M(P)$ has at least $\sqrt{r}$ different rows, so that the deterministic protocol $P$ has to consist of at least $\sqrt{r}$ messages. ■

In the proof of Theorem 2.1 we have assumed that the probability of the output $*$ is bounded by $\frac{1}{2}$ for every input. Since for Las Vegas computing models the size of the upper bound on unsuccess is not essential, one can use any upper bound $\varepsilon < 1$ of computing $*$ in the definition of Las Vegas protocols. One can easily observe that the exchange of the upper bound $\frac{1}{2}$ on unsuccess for an arbitrary upper bound $\varepsilon$, $0 < \varepsilon < 1$, would result in the fact that the matrix $M(P)$ has at least $r^{1-\varepsilon}$ different rows. So, in this general case the claim of Theorem 2.1 would be $\text{lvcc}_1(f) \geq (1 - \varepsilon) \cdot \text{cc}_1(f)$.

To show that the lower bound of Theorem 2.1 cannot be improved we consider the language

$$L = \{xy \in \{0,1\}^* \mid |x| = |y| \text{ and if } y = 0^i 1 z, \text{ then } x_{i+1} = 1, \text{ for } 0 \leq i < |x|\}.$$

Remember, that for every $n \in \mathbb{N}$, $h_n(L)$ is a Boolean function of $n$ variables defined by $h_n(L)(\alpha) = 1$ iff $\alpha \in L \cap \{0,1\}^n$.

THEOREM 2.2. For every positive integer $n$,

(i) $\text{cc}_1(h_{4n}(L)) = 2n$,

(ii) $\text{lvcc}_1(h_{4n}(L)) = n$, and

(iii) $\lceil \log_2 2n \rceil \leq \text{svncc}_1(h_{4n}(L)) \leq \lceil \log_2 2n \rceil + 1$.

*Proof.* It is well known that, for every Boolean function $f$, $\text{cc}_1(f)$ is equal to the logarithm of the number of different rows in $M(f)$ (see [7, 9]). Since there are no two identical rows in $M(h_{4n}(L))$ and $M(h_{4n}(L))$ has $2^{2n}$ rows, the result (i) follows.

We obtain a Las Vegas protocol for $h_{4n}(L)$ communicating $n$ bits as follows. The first computer flips an unbiased coin and sends accordingly the first respectively the second half of its input. Obviously the second computer is now able to determine the result with probability $\frac{1}{2}$. Because of (i) and Theorem 2.1 there is no better protocol.

The fact $\text{svncc}_1(h_{4n}(L)) \geq \lceil \log_2 2n \rceil$ is obvious because $\text{svncc}_1(h_{4n}(L)) \geq \lceil \log_2(\text{cc}_1(f)) \rceil$ for every $f$ [19]. On the other hand, $\lceil \log_2 2n \rceil + 1$ bits suffice for a self-verifying protocol, if processor $C_I$ guesses

a position $j \in \{1, \ldots, 2n\}$ and sends the binary code of $j$ and the bit $x_j$ to $C_{II}$. $C_{II}$ knows the crucial bit position and gives a binding answer if position $j$ matches. ■

Thus, Theorem 2.2 shows not only the optimality of the lower bound of Theorem 2.1, but also an exponential gap between determinism and self-verifying nondeterminism. Note that this exponential gap cannot be improved because $\text{ncc}_1(f) \leq 2^{\text{cc}_1(f)}$ for every Boolean function $f$ [7, 19]. This result contrasts to the at most quadratic gap between determinism and self-verifying nondeterminism for two-way communication complexity. To show also an exponential gap between nondeterminism and self-verifying nondeterminism, it suffices to consider the language $ID^C = \{xy \in \{0,1\}^* \mid x \neq y, |x| = |y|\}$.

OBSERVATION 2.3. *For every positive integer* $n$,

(i) $\text{ncc}_1(h_{2n}(ID^C)) \leq \lceil \log_2 n \rceil + 1$, *and*

(ii) $\text{svncc}_1(h_{2n}(ID^C)) = n$.

*Proof.* (i) is obvious because $C_I$ can guess a position $j$ in which the inputs of $C_I$ and $C_{II}$ differ. So, the message consists of the binary code of $j$ and of the $j$th bit of the input of $C_I$.

Obviously, for even input lengths the identity language $ID = \{xx \mid x \in \{0,1\}^*\}$ is the complement of $ID^C$. Since it is well known that $\text{ncc}_1(h_{2n}(ID)) = n$ (see, for example, [7, 8]) and

$$\text{svncc}_1(f) \geq \max\{\text{ncc}_1(f), \text{ncc}_1(f^C)\}$$

for every function $f$, equality (ii) follows. ■

## 3. OBDDS

In what follows we apply Theorem 2.1 to get a polynomial relationship between Las Vegas and determinism for OBDDs. OBDDs [11, 12] are highly restricted branching programs [13]. A branching program is a directed acyclic graph with one source and two sinks labelled by Boolean constants 0 and 1. The non-sink nodes are labelled by Boolean variables and have two outgoing edges labelled by 0 and 1. The computation of a branching program $A$ on an input $a = a_1 a_2 \ldots a_n$ starts at the source. At an inner node labelled by $x_i$ the outgoing edge with the label $a_i$ is chosen. The label of the sink that is reached defines $f_A(a)$ for the Boolean function $f_A$ computed by $A$. An OBDD is a branching program that satisfies the following restrictions:

(i) for every input variable $x$ and for every directed path $P$ of $A$, $P$ contains at most one node labelled by $x$, and

(ii) there exists an ordering $X_A = x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ of the input variables $x_1, x_2, \ldots, x_n$ such that the sequence of the labels of nodes on every directed path of $A$ is a subsequence of $X_A$.

An OBDD $A$ is called *levelled* if every path from the source to a sink has length $n$; i.e., the nodes of $A$ can be partitioned into $n+1$ disjoint sets $l_0(A), l_1(A), \ldots, l_n(A)$, where nodes in the $i$th level $l_i(A)$ have distance $i$ from the source, and all nodes of $l_i(A)$ have the same label. The *width* of a levelled OBDD $A$ is

$$\text{width}(A) = \max\{|l_i(A)| \mid i = 1, \ldots, n\}.$$

The most important complexity measure for an OBDD is its *size*, namely the number of its nodes. For any Boolean function $f$, we denote by *size*$(f)$ [*lev-size*$(f)$] the size of the best [levelled] OBDD for $f$.

In the literature (see, e.g., [15]) randomized branching programs and OBDDs have been investigated. We consider for the first time Las Vegas OBDDs (LV-OBDDs). The extension of OBDDs to LV-OBDDs is straightforward. One adds a new sink labelled by $*$ ("don't know"). For any nonsink node $v$ one allows several edges $(v, u_1), \ldots, (v, u_k)$, $k \geq 1$, labelled by the same Boolean constant. To each edge $(v, u_i)$ the probability $\text{prob}(v, u_i)$ is assigned. The only requirements are $0 < \text{prob}(v, u_i) \leq 1$ for every $i$ and $\sum_{i=1}^{k} \text{prob}(v, u_i) = 1$. The meaning is that during the computation of a LV-OBDD at each inner node

labelled by $x_i$ one of the edges labelled by $a_i$ is chosen with probability given by function prob. So, the probability of a path $s_1, s_2, \ldots, s_n, s_{n+1}$ is $\Pi_{i=1}^{n} \text{prob}(s_i, s_{i+1})$.

We say that a LV-OBDD $A$ computes a Boolean function $f$ if for every input $a$ the probability of reaching the sink labelled by $f(a)$ is at least $\frac{1}{2}$ and the probability of reaching the sink labelled by $\overline{f(a)}$ is 0. Let *LV-size*$(f)$ (*LV-lev-size*$(f)$) denote the size of the best Las Vegas (levelled) OBDD for $f$.

To establish the relationship between Las Vegas and determinism for OBDDs we present a few simple facts. In what follows we need to consider one-way protocols for arbitrary input partitions. A partition of a set $X = \{x_1, \ldots, x_n\}$, $n \in \mathbb{N}$, of input variables is any pair $\Pi = (\Pi_I, \Pi_{II})$ where $\Pi_I \cup \Pi_{II} = X$ and $\Pi_I \cap \Pi_{II} = \emptyset$. A one-way protocol according to a partition $\Pi = (\Pi_I, \Pi_{II})$ is the usual one-way protocol; the only difference is that the computer $C_I$ obtains values of variables from $\Pi_I$ and $C_{II}$ obtains values of variables from $\Pi_{II}$. Let $mc(D)$ denote the number of different messages of a deterministic one-way protocol $D$. Let $cc_1(f, \Pi)$ be the one-way communication complexity of the best protocol computing $f$ according to $\Pi$, and let $mc(f, \Pi) = \min\{\text{mc}(D) \mid D \text{ is a one-way protocol computing } f \text{ according to } \Pi\}$.

The following statement is implicit in [7, 20].

FACT 3.1. *Let $f$ be a Boolean function defined over a set of Boolean variables $X = \{x_1, \ldots, x_n\}$, $n \in \mathbb{N}$. Let $A$ be a levelled OBDD computing $f$ with $X_A = x_{i_1}, x_{i_2}, \ldots, x_{i_n}$. Then, for every $j = 1, 2, \ldots, n$, there is a one-way protocol $D_j$ computing $f$ according to*

$$\Pi_j = \left(\{x_{i_1}, \ldots, x_{i_j}\}, \{x_{i_{j+1}}, \ldots, x_{i_n}\}\right)$$

*with*

$$\text{mc}(D_j) = |l_j(A)|.$$

FACT 3.2. *Let $f$ be a Boolean function defined over a set of Boolean variables $X = \{x_1, \ldots, x_n\}$, $n \in \mathbb{N}$. Let $Y = x_{i_1}, \ldots, x_{i_n}$ be a permutation of $x_1, \ldots, x_n$. Let, for every $j = 1, \ldots, n$, $D_j$ be a one-way protocol computing $f$ according to $\Pi_j = (\{x_{i_1}, \ldots, x_{i_j}\}, \{x_{i_{j+1}}, \ldots, x_{i_n}\})$. Then there exists a levelled OBDD $A$ computing $f$ with $X_A = Y$ and*

$$|l_j(A)| \leq \text{mc}(D_j)$$

*for every $j \in \{1, \ldots, n-1\}$.*

*Proof.* For every $j \in \{1, \ldots, n\}$, $\text{mc}(D_j)$ is greater than or equal to the number of different rows of the communication matrix $M_j(f)$ obtained from $f$ by using the partition $\Pi_j$ of input variables [9, 19]. The number of different rows of $M_j(f)$ is exactly the number $r_j$ of different subfunctions of $f$ obtained by fixing $x_{i_1}, \ldots, x_{i_j}$ to arbitrary values. One can then construct a levelled OBDD for $f$, where the $j$th level contains exactly $r_j$ nodes [20]. ■

COROLLARY 3.1. *Let $f$ be a Boolean function defined over the set of Boolean variables $X = \{x_1, \ldots, x_n\}$, $n \in \mathbb{N}$. Let $A$ be a size-optimal levelled OBDD computing $f$ with $X_A = x_{i_1}, \ldots, x_{i_n}$. Then*

$$\text{size}(A) = \text{lev-size}(f) = \sum_{j=1}^{n} \text{mc}(f, \Pi_j) + 1,$$

*where $\Pi_j = (\{x_{i_1}, \ldots, x_{i_j}\}, \{x_{i_{j+1}}, \ldots, x_{i_n}\})$.*

Note that Fact 3.1 is true also for nondeterministic and randomized versions of OBDDs and one-way protocols. So, we obtain the following result.

THEOREM 3.1. *For every Boolean function $f$*

$$\text{LV-lev-size}(f) \geq \sqrt{\text{lev-size}(f)}.$$

*Proof.* Let $A$ be a size-optimal levelled LV-OBDD computing $f : \{0,1\}^n \to \{0,1\}$. Let $f$ be defined over $X = \{x_1, \ldots, x_n\}$, and $X_A = x_{i_1}, \ldots, x_{i_n}$ for a permutation $(i_1, \ldots, i_n)$ of $(1, \ldots, n)$. For every $j \in \{1, \ldots, n\}$, let $\Pi_j = (\{x_{i_1}, \ldots, x_{i_j}\}, \{x_{i_{j+1}}, \ldots, x_{i_n}\})$. By considering a randomized version of Fact 3.1 we have, for every $j \in \{1, \ldots, n-1\}$, a Las Vegas one-way protocol $B_j$ computing $f$ according to $\Pi_j$ with $\mathrm{mc}(B_j) = |l_j(A)|$. According to the proof of Theorem 2.1 we have

$$\mathrm{mc}(B_j) \geq \sqrt{\mathrm{mc}(f, \Pi_j)}$$

for every $j \in \{1, \ldots, n-1\}$. Because of Fact 3.2 there is a levelled OBDD $C$ computing $f$ with $X_C = X_A$ and $|l_j(C)| = \mathrm{mc}(f, \Pi_j)$. From Corollary 3.1 it then follows that

$$\mathrm{size}(A) = \sum_{j=0}^{n} |l_j(A)| = 1 + \sum_{j=1}^{n} \mathrm{mc}(B_j) \geq 1 + \sum_{j=1}^{n} \sqrt{\mathrm{mc}(f, \Pi_j)} = 1 + \sum_{j=1}^{n} \sqrt{|l_j(C)|}.$$

Since $\mathrm{size}(C) = 1 + \sum_{j=1}^{n} |l_j(C)|$ and $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for any positive integers $a$ and $b$, we obtain

$$\text{LV-lev-size}(f) = \mathrm{size}(A) \geq \sqrt{1 + \sum_{j=1}^{n} |l_j(C)|} = \sqrt{\mathrm{size}(C)} \geq \sqrt{\text{lev-size}(f)}. \qquad \blacksquare$$

In the case of general OBDDs we have a weaker result than for levelled OBDDs. But also in this case there is no exponential gap between determinism and Las Vegas for the size of OBDDs.

THEOREM 3.2. *For any Boolean function $f$ of $n$ variables*

$$\text{LV-size}(f) \geq \sqrt{\mathrm{size}(f)/n}.$$

*Proof.* Let $X = \{x_1, \ldots, x_n\}$ be the set of input variables of $f$. Let $A$ be a size-optimal LV-OBDD for $f$ (i.e., $\mathrm{size}(A) = \text{LV-size}(f)$). Let $X_A$ be the variable ordering of $A$. Without loss of generality we assume $X_A = x_1, x_2, \ldots, x_n$. Let $B$ be the size-optimal OBDD for $f$ among all OBDDs with the variable ordering $X_A$. Since $\mathrm{size}(B) \geq \mathrm{size}(f)$, it is sufficient to prove that $\mathrm{size}(A) \geq \sqrt{\mathrm{size}(B)/n}$.

Let, for every $i = 1, \ldots, n$, $\Pi_i = (\{x_1, \ldots, x_{i-1}\}, \{x_i, \ldots, x_n\})$. We consider a cut of $A$ according to $\Pi_i$ as the cut into two parts where one part $L_i$ contains nodes with label in $\{x_1, x_2, \ldots, x_{i-1}\}$ and the second part $R_i$ contains nodes with label in $\{x_i, x_{i+1}, \ldots, x_n\}$. Obviously, for each $\Pi_i$ there is a one-way Las Vegas protocol $D_i$ computing $f$ according to $\Pi_i$ within message complexity at most $|R_i|$. (The messages of $D_i$ are names of nodes of $R_i$ reached after reading the input part corresponding to the variables $x_1, x_2, \ldots, x_{i-1}$.) Moreover,

$$\text{LV-size}(f) = \mathrm{size}(A) \geq |R_i| \tag{1}$$

for every $i \in \{1, \ldots, n\}$. According to the proof of Theorem 2.1 we have

$$|R_i| \geq \sqrt{\mathrm{mc}(f, \Pi_i)} \tag{2}$$

for every $i \in \{1, \ldots, n\}$. Since $B$ is a size-optimal OBDD with ordering $X_A$, the number $\alpha_i$ of nodes of $B$ with label $x_i$ is a lower bound for $\mathrm{mc}(f, \Pi_i)$ [20]. Choose some $i \in \{1, \ldots, n\}$ with $\alpha_i \geq \mathrm{size}(B)/n$; then we have

$$\mathrm{mc}(f, \Pi_i) \geq \mathrm{size}(B)/n. \tag{3}$$

Concatenating (1), (2), and (3) we get

$$\text{LV-size}(f) = \mathrm{size}(A) \geq |R_j| \geq \sqrt{\mathrm{mc}(f, \Pi_j)} \geq \sqrt{\mathrm{size}(B)/n}. \qquad \blacksquare$$

## 4. FINITE AUTOMATA

The main goal of this section is to show an at most quadratic gap between Las Vegas and determinism.

The idea of our approach is to find a strong connection between the number of messages of one-way protocols and the number of states of finite automata in such a way that Theorem 2.1 can be applied. Such a connection between one-way protocols and finite automata has been observed already in [21] in the form

$$\mathrm{cc}_1(h_n(L)) \leq \lceil \log_2(\mathrm{s}(L)) \rceil$$

for every regular language $L \subseteq \{0,1\}^*$ and every $n \in N$. Obviously this relation holds in the nondeterministic and randomized cases, too. More precisely, the number of different messages of the best one-way protocol is a lower bound on the number of states of finite automata. Unfortunately the difference between these two complexity measures may be arbitrarily large because communication complexity is defined in terms of a non-uniform computing model, whereas automata form a uniform computing model: Consider, for instance unary languages $L$ where we always have $\mathrm{cc}_1(h_n(L)) \leq 1$ but for any constant $c$ we have a unary language $L'$ with $\mathrm{s}(L') > c$. To overcome this difficulty we introduce one-way uniform protocols:

DEFINITION 4.1. Let $\Sigma$ be an alphabet and let $L \subseteq \Sigma^*$. *A one-way uniform protocol over* $\Sigma$ is a pair $D = \langle \Phi, \varphi \rangle$, where:

(i) $\Phi : \Sigma^* \to \{0,1\}^*$ is a function with the prefix freeness property ($\Phi(x)$ is no proper prefix of $\Phi(y)$ for any $x, y \in \Sigma^*$), and

(ii) $\varphi : \{0,1\}^* \times \{0,1\}^* \to \{accept, reject\}$ is a function.

We say that $D = \langle \Phi, \varphi \rangle$ *accepts* $L$, $L(D) = L$, if for all $x, y \in \Sigma^*$:

$$\varphi(y, \Phi(x)) = accept \Leftrightarrow xy \in L.$$

The *message complexity of the protocol* $D$ is

$$mc(D) = |\{\Phi(x) \mid x \in \Sigma^*\}|$$

and we define the *message complexity of* $L$ as

$$mc(L) = \min\{mc(D) \mid D \text{ is a one-way uniform protocol accepting } L\}.$$

Finally we consider special infinite communication matrices:

DEFINITION 4.2. Let $\Sigma$ be an alphabet and let $L \subseteq \Sigma^*$. We define the infinite Boolean matrix $M_L = (a_{uv})_{u,v \in \Sigma^*}$ so that

$$a_{uv} = 1 \Leftrightarrow uv \in L.$$

Let $\mathrm{row}_L$ be the number of different rows of $M_L$.

Now, we can formulate the crucial observation claiming that the message complexity of a regular language $L$ is the same complexity measure as the size of the minimal finite automaton for $L$.

LEMMA 4.1. *For every regular language* $L$ *over an alphabet* $\Sigma$,

$$\mathrm{s}(L) = \mathrm{mc}(L) = \mathrm{row}_L.$$

*Proof.* The equality $s(L) = row_L$ is just a reformulation of the Myhill–Nerode theorem, because $row_L$ is exactly the index of the right invariant relation on $\Sigma^*$ according to $L$.

Since $s(L)$ is finite for every regular language $L$, and the number of different rows of a communication matrix is equal to the number of different messages used by the best one-way protocol computing this matrix (for this well-known fact see, for instance [7–9]), and we obtain $mc(L) = row_L$. ■

Now, we are ready to formulate our main result.

THEOREM 4.1. *For every regular language $L$,*

$$lvs(L) \geq \sqrt{s(L)}.$$

*Proof.* Let $L$ be a regular language over an alphabet $\Sigma$. Since $row_L$ is finite, one can easily find a finite submatrix $M$ of $M_L$ with $row_L = s(L)$ different rows. Let $f$ be the finite function of two arguments that correspond to $M$. Then the optimal one-way protocol for $f$ uses exactly $row_L$ different messages. Let $A$ be a LVFA for $L$ with $lvs(L)$ states. In the obvious way, this automaton induces a one-way Las Vegas protocol for the function $f$ that uses $lvs(L)$ different messages. In the proof of Theorem 2.1 it was shown that in this situation we must have $lvs(L) \geq \sqrt{row_L}$. By Lemma 4.1 the result follows. ■

The language $L_k = \{w \in \{0,1\}^* \mid w = u1v \text{ and } |v| = k-1\}$ is a well-known example of a language producing an exponential gap between $s(L)$ and $ns(L)$ [22]. We use $L_k$ to show that Theorem 4.1 cannot be improved. But first we give the following useful observation expressing the typical property of self-verifying nondeterminism.

OBSERVATION 4.2. *For any regular language $L$:*

$$\max\{ns(L), ns(L^C)\} \leq svns(L) \leq 1 + ns(L) + ns(L^C).$$

*Proof.* Every SNFA $A$ can be changed to an NFA $B$ by adding the neutral states to the set of rejecting states. Obviously $L(A) = L(B)$ and $s(A) = s(B)$. If one takes the rejecting states of $A$ as accepting ones of an NFA $C$, and the accepting and neutral states as the rejecting ones of $C$, then $L(C) = (L(A))^C$. So, $svns(L) \geq \max\{ns(L), ns(L^C)\}$.

Let $E$, and $F$ be NFAs such that $L(E) = (L(F))^C$. A SNFA $D$ can be constructed as follows. $D$ connects a new initial state via $\varepsilon$-moves to the initial states of $E$ and $F$. Finally $D$ chooses as accepting states the set of accepting states of $E$, as rejecting states the set of accepting states of $F$, and makes the remaining final states of $E$ and $F$ neutral. Then obviously $L(E) = L(D)$. ■

THEOREM 4.2. *For every positive integer $k$,*

(i) $s(L_k) = 2^k$,
(ii) $lvs(L_k) \leq 4 \cdot 2^{k/2} = O(\sqrt{s(L_k)})$,
(iii) $svns(L_k) \leq 2k + 3$, *and*
(iv) $ns(L_k) = k + 1 = ns(L_k^C)$.

*Proof.* (i) and (ii) are well-known facts. (iii) follows immediately from Observation 4.2 and from (iv). To show (ii) we consider the following strategy of a LVFA $A$. The computation of $A$ starts by randomly guessing whether the important bit (the $k$th bit from the end) is on an even or odd bit position. Now, one can easily construct a deterministic FA of $2^{k/2+1}$ states accepting $L_k^{odd} = \{w \in \{0,1\}^* \mid w = u1v, |v| = k-1$, and $|u|$ is odd$\}$ or $L_k^{even} = \{w \in \{0,1\}^* \mid w = u1v, |v| = k-1$, and $|u|$ is even$\}$. ■

Again we see that self-verifying nondeterminism may be much more powerful than determinism (resp. Las Vegas). If one wishes to demonstrate a large difference between self-verifying nondeterminism and nondeterminism, one has to choose a regular language with a large difference between $ns(L)$ and $ns(L^C)$. We consider for every $m \in N$ the language

$$U_m = \{u0v1w, u1v0w \mid |v| = m-1, uvw \in \{0,1\}^*\}.$$

OBSERVATION 4.3. *For every* $m \in N$

(i) $\text{ns}(U_m) \leq 2m + 2$,

(ii) $\text{ns}(U_m^C) \geq 2^m$, *and*

(iii) $\text{svns}((U_m)) \geq 2^m$.

*Proof.* (i) is obvious. Since $\{xx \mid x \in \{0,1\}^m\} = U_m^C \cap \{0,1\}^{2m}$ every NFA accepting $U_m^C$ must be in different states after reading two different words $y, z \in \{0,1\}^m$. Since $|\{0,1\}^m| = 2^m$ we obtain (ii). (iii) is a direct consequence of (ii) and Observation 4.2. ■

## ACKNOWLEDGMENTS

We are indebted to Noam Nisan and to an unknown referee who pointed out to us the possibility of applying Theorem 2.1 for proving the polynomial relation between Las Vegas and determinism for OBDDs. We thank the referees for valuable comments and suggestions that have essentially contributed to the improvement of the presentation of this paper.

## REFERENCES

1. Babai, L., Frankl, P., and Simon, J. (1986), Complexity classes in communication complexity theory, *in* "Proceedings, 27th IEEE Symposium on the Foundations of Computer Science," pp. 337–347.
2. Mehlhorn, K., and Schmidt, E. (1982), Las Vegas is better than determinism in VLSI and distributed computing, *in* "Proceedings, 14th ACM Symposium on Theory of Computing," San Francisco, CA, pp. 330–337.
3. Savitch, W. J. (1970), Relationships between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* **4**,177–192.
4. Dietzfelbinger, M., Kutylowski, M., and Reischuk, R. (1994), Exact lower bounds for computing Boolean functions on CREW PRAMs, *J. Comput. System Sci.* **48**, 231–254.
5. Bovet, D. P., and Crescenzi, P. (1994), "Introduction to the Theory of Complexity," Prentice Hall, New York.
6. Yao, A. C. (1981), Some complexity questions related to distributed computing, *in* "Proceedings, 11th ACM Symposium on Theory of Computing," pp. 308–311.
7. Hromkovič, J. (1997), "Communication Complexity and Parallel Computing," Springer-Verlag, Berlin.
8. Kushilevitz, E., and Nisan, N. (1997), "Communication Complexity," Cambridge Univ. Press, Cambridge, UK.
9. Aho, A. V., Hopcroft, J. E., and Yannakakis, M. (1983), On notions of information transfer in VLSI circuits, *in* "Proceedings, 15th ACM Symposium on Theory of Computing," pp. 133–139.
10. Hromkovič, J., and Schnitger, G. (1996), On the power of the number of advice bits in nondeterministic computations, *in* "Proceedings, 28th ACM Symposium on Theory of Computing," pp. 551–560.
11. Bryant, R. E. (1985), Symbolic manipulation of Boolean functions using a graphical representation, *in* "22nd ACM/IEEE Design Automation Conf.," pp. 688–694.
12. Bryant, R. E. (1986), Graph-based algorithms for Boolean function manipulation, *IEEE Trans. Comput.* **35**, 677–691.
13. Pudlák, P., and Žák, S. (1983), "Space Complexity of Computation," Technical Report, Prague.
14. Wegener, I. (1987), "The Complexity of Boolean Functions," Wiley-Teubner Series in Computer Science, Wiley, New York and Teubner, Stuttgart.
15. Sauerhoff, M. (1998), Lower bounds for randomized read-k-times branching programs, *in* "Proceedings, 15th Symposium on Theoretical Aspects in Computer Science," Lecture Notes in Computer Science, vol. 1373, pp. 105–115, Springer-Verlag, Berlin.
16. Klauck, H. (2000),On quantum und probabilistic communication: Las Vegas and one-way protocols, *in* "Proceedings, 32th ACM Symposium on Theory of Computing," pp. 644–651.
17. Newman, I. (1991), Private vs. common random bits in communication complexity, *Inform. Process. Lett.* **39**, 301–315.
18. Csiszar, I., and Körner, J. (1986), "Information Theory: Coding Theorems for Discrete Memoryless Systems," Academic Press, New York.
19. Papadimitriou, Ch., and Sipser, M. (1984), Communication complexity, *J. Comput. System Sci.* **28**, 260–269.
20. Sieling, D., and Wegener, I. (1993), NC-algorithms for operations on binary decision diagrams, *Parallel Process. Lett.* **3**, 3–12.
21. Hromkovič, J. (1986), Relation between Chomsky hierarchy and communication complexity hierarchy, *Acta Math. Univ. Comenian (N.S.)* **48–49**, 311–317.
22. Meyer, A. R., and Fischer, M. J. (1971), Economies of description by automata, grammars and formal systems, *in* "Proceedings, 12th IEEE Symposium on Switching and Automata Theory," pp. 188–191.